Student Practice Sessions Modeled as ICAP Activity Silos

Adam M. Gaweda, Collin F. Lynch
North Carolina State University
Raleigh, NC, USA
agaweda,cflynch@ncsu.edu

ABSTRACT
There are a number of novel exercise types that students can utilize while learning Computer Science, each with its own level of complexity and interaction as outlined by the ICAP Framework [10]. Some are *Interactive*, like solving coding problems; *Constructive*, like explaining code; *Active*, like retyping source code; and *Passive*, like reviewing slides. To date, there has been little research on how students vary their study and engagement habits by exercise type and when they do so. In this paper, we present our findings on student activity sequences from an online professional development course. We isolated student activities into sessions and then produced activity transition visualizations to compare the behavior of students who complete the course to those who do not. We then used multiple factor analyses to examine how students transition from one type of activity to the next. From this analysis we identified *platform silos* in student's work. We further expand this concept to the presence of *activity silos* grouping by type. We find that this siloing behavior is consistent in both completers and non-completers but is weaker for the latter group. Finally, we discuss our findings and how instructors and researchers may use this information to ensure that students show persistence through practice.

Keywords
novel exercises, ICAP framework, study sessions, activity sequences, platform silos, activity silos, student modeling

1. INTRODUCTION
CS Education have introduced a number of novel exercise types to better scaffold students' experiences. These include retyping source code [14], arranging scrambled code fragments (Parsons Puzzles) [21], debugging provided code [8], predicting output [26], fill in the blanks [4], self-explanation [4], and small scale coding exercises [2, 12]. Each of these exercise types can also be mapped onto the ICAP framework [10]. This framework defines four categories of instructional activities based upon students' level of engagement: Interactive, Constructive, Active, and Passive. *Passive learning* includes reading static course materials or watching lecture videos. *Active learning* is described as rehearsing or copying solution steps. *Constructive learning* includes self-explanation of content or creation of novel externalized outputs like summaries. Finally, *Interactive learning* involves directly engaging with a peer, agent, or instructor to explore information and receive feedback which can be expanded upon.

Adam Gaweda and Collin Lynch "Student Practice Sessions Modeled as ICAP Activity Silos". 2021. In: Proceedings of The 14th International Conference on Educational Data Mining (EDM21). International Educational Data Mining Society, 595-601. https://educationaldatamining.org/edm2021/
EDM '21 June 29 - July 02 2021, Paris, France

While these exercise types have made their way into classrooms there is little evidence of how these types of engagement interact with one-another. Traditional intervention studies have focused on the overall impact of one or more exercise types [20, 19, 14, 8], or on the automated selection/recommendation of future excercises based upon a student model [27], but not on how students work with or across them in the absence of guidance. Nor has this recommendation work been extended to nontraditional learning contexts. Absent an understanding of how students orchestrate multiple interaction modes we face challenges in scaffolding effective learning opportunities and in evaluating the impact of novel learning environments. Providing students with ineffective, or overly complex learning opportunities risks trapping them in a fail/skip practice cycle that would inhibit any functional learning gains [17].

In this paper we report our investigation of how students direct their practice of CS concepts when presented with a set of options. Our study was conducted in the context of an online professional development course for Python programming. This course is part of a research study funded by the Department of Labor to create novel learning pathways for existing technical professionals to move into AI and Data-Science areas. We extracted students' practice/study sessions and analyzed the activity transitions within each session. We then analyzed these activity sequences to answer the following research questions (RQs):

RQ1 Can we replicate the existence of *platform silos* introduced in [1] with a new dataset?
RQ2 Are there common activity transitions between students?
RQ3 How do activity sequences connect with the ICAP Framework?
RQ4 How do the practice sessions of completers of the course differ from non-completers?

To answer these questions, we first produced and analyzed a set of activity transition diagrams for students in the course comparing those who completed the course to those who did not. Through this analysis, we confirmed the presence of *platform silos* which we extend this notion to include *activity silos*, where students primarily focus on a single mode of engagement (consistent with ICAP) during a given practice session. When students did transition between modes, it was only to move up the ICAP chain and never to 'downgrade' to a lower level of engagement. We support our findings through two different factor analyses, which help explain the 42-62% of variance between the sessions. From our results, we found ICAP categories isolated into individual sessions, as well as LMS content consumption and quiz taking.

2. BACKGROUND

2.1 ICAP Framework

The ICAP Framework seeks to classify the modes of student engagement while engaging in learning activities [10]. These categories, *Interactive, Constructive, Active* and *Passive* respectivel. *Passive* engagement includes activities such as reading a text or observing a video. *Active* engagement includes rehearsing steps or copying solutions. *Constructive* engagement includes self-explanation or comparing and contrasting materials. Finally, *Interactive* engagement includes responding and interacting with an agent, system, or another person. Tthe framework is hierarchical, suggesting I > C > A > P, or that activities with higher levels of engagement promote the greatest levels of learning.

Chi and Wylie present a literature review of several empirical studies supporting their ICAP hypothesis [10]. The first study consisted of all four modes of engagement in materials science and showed learning improved significantly at a rate of 8-10% per mode. They then present two studies which used active, passive, and constructive modes in evolutionary biology and plate tectonics. Finally, Chi and Wylie present comparisons of two modes across note taking, concept mapping, and self-explanation. In each of these studies, the results again showed that higher modes of engagement had higher learning gains.

Chi and Wylie's work, as well as our own personal communications with Chi [9], note that identifying the mode of a particular activity can be a non-trivial task. For example, a toy example task could be presenting steps toward making a peanut butter and jelly sandwich given randomly shuffled segments of instructions. This task could be constructed as an *active* exercise if the student already knows the recipe and the task is simply picking the appropriate sequence from the list of steps. However, if the student had not learned the appropriate order, then the task of figuring out the right sequence would be construed as a *constructive* exercise. Computer Science has a similar task, known as Parsons Puzzles, that mirrors this toy example, discussed in more detail in the following section.

2.2 Novel Exercise Types

In this section, we describe eight different exercise type studied in Computer Science education, provide some research background on the exercise type, and justifications for which ICAP mode we will classify them as for of our study.

2.2.1 Typing Exercises

Typing Exercises (TE) require students to retype source code that has been presented to them [14]. Typing Exercises can be used as *active* learning activities under the ICAP Framework as they require students to retype verbatim the code presented to them. Previously, we presented images of source code and showed that self-selected students that completed optional typing exercises earned higher course grades and submitted less code with build failures. Leinonen et. al. also presented optional typing exercises to students before programming tasks [19], but were not able to find the same results as ours. However their study only lasted two weeks and many of their selected participants did not attempt the exercises at all.

2.2.2 Fill in the Blank

Fill in the Blank ($FitB$) exercises remove a small portion of code from a snippet and asks students to 'fill in' the blank. Students need to have an understanding of the snippet as a whole to deduce what needs to be included at a particular blank location. Reviewing incomplete worked examples reduces ineffective self-explanations and enhances the transfer of learned materials [4, 5]. Based on the results from Atkinson et. al., we consider FitB-style exercises to require lower levels of engagement than self-explanation, and thus classify them as an *active* ICAP mode.

2.2.3 Parsons Puzzles

Parsons Puzzles (PP) present snippets of code that have been separated into segments and then shuffled in order [21]. Students are then tasked with placing the segments back into the correct order. While Parsons Puzzles are helpful for learning how to structure code, performance on Parsons Puzzles has not been shown to correlate with students' ability to read or trace code [11, 20] and 'distractor' variants are not beneficial to young learners [13, 16]. These findings further support our research goals, as not every exercise type may be beneficial for learning all the technical skills necessary for Computer Science.

Chi and Wylie's definition of constructive modes of engagement include "learners generate or produce additional externalized outputs... beyond what is provided". As mentioned in the previous section, Parsons Puzzles are similar to our toy peanut butter and jelly exercise. While Parsons Puzzles could be construed as *active*, students may not fully comprehend the appropriate order of code syntax and must figure out the right sequence as part of the exercise. In our communications with Chi about Parsons Puzzles, Chi states that determining the particular ICAP mode for an activity can be non-trivial [9]. 'If a student already [knows] the recipe, then re-ordering it is just picking out the sequencing, guided by the sequence information [they] already know' then it is *active*. However, 'if the student [has] not learned the order from some other source, and you are asking her to figure out the right sequence', it is a *constructive* exercise. Since novices may not have proficiency at the time of the exercise, we elect to use the upper bound ICAP mode and consider Parsons Puzzles as a *constructive* learning activity.

2.2.4 Output Prediction

Output Prediction (OP), also known as variable tracing, exercises ask students to analyze code and then state the expected outputs of code execution or the expected value of a variable as the code progresses. Often, variable tracing is done during conditional and loop instruction to demonstrate how the values of the variables change after each iteration. Like Parsons Puzzles, OP-style exercises require students to process code snippets and externalize their expected outputs. Thus, we consider output prediction as another *constructive* learning activity.

2.2.5 Self-Explanation

Self-explanation (SE) exercises present students with source code and ask the student to explain how the code operates, describe the overall efficiency of the code, or create a documentation string to appear as a comment for the program or function. These are open-ended exercises that are subjective in nature and are considered to be *constructive* [10]. However, Chi and Wylie do note that students' may treat the self-explanation activity as *active* if "the student's self-explanation is verbatim to what was read". However, novices may struggle with reading and evaluating programming code in a linear fashion, focusing more on what each line of code did, rather than how each line interacted with each other, or in general produce poor explanations [25, 5]. While *constructive* SE activities may produce higher learning gains than lower-level modes, they may also not be the most appropriate activity for students who are struggling, Thus, similar to our decisions for Parsons Puzzles, we use the upper bound to classify self-explanation as a *constructive* learning activity.

2.2.6 Find and Fix the Bug

Resolving errors, or debugging, is one of the first hurdles students encounter when learning to program [3]. Once they find that an error has occurred, it will be necessary for them to resolve it before addressing any remaining subgoals for their solution. For the purposes of our study, we separated debugging into two separate activities - Find the Bug (FnB) and Fix the Bug (FxB). Find the Bug exercises present students with code that contains a common misconception for novices. Instead of resolving the error, students are asked to highlight the area of code where this error exists. Though the ICAP framework considers highlighting text as an *active* learning activity, Chi and Wylie define *constructive* behaviors as requiring some level of "inference", or adding in additional detail or qualification. Since students must assess if a line of code is 'correct' or not, they are producing qualifications and therefore, we elect to label FnB exercises as *constructive*.

Fix the Bug exercises follow the natural progression of debugging tasks by requiring students to resolve broken code[8]. FxB activities could be considered *constructive* or *interactive* depending on the context. Similar to FnB exercises, Chi and Wylie include 'repairing' as a *constructive* behavior. However, FxB-style exercises can also be *interactive* because students often rely on the code interpreter's feedback during the debugging process. Fixing one error may produce new errors with new feedback, or the repair made by the student could be incorrect. Thus, we again choose use the upper bound to label FxB exercises as *interactive*.

2.2.7 Coding Exercises

The final exercise type we used in our study is the de facto standard of introductory CS courses - the Coding Exercise (CE). While Computer Science is more than programming, coding exercises are often used by instructors as graded course material for students to demonstrate their understanding of the current course topic. There has been work on the use of 'many-small programs' and 'simple syntax exercises', which simply require students to complete small-scale coding exercises to become familiar with the a particular implementation before utilizing it as part of larger-scale problems [2, 12]. In both cases, completing these smaller-scale programming exercises improved student performance and yielded happier students. Students often rely on feedback from the interpreter as they construct their solutions and more than likely need to debug their own work during this process. Thus, we consider Coding Exercises as an *interactive* learning activity.

2.3 Student Modeling and Activity Mining

Seshadri et. al. analyzed how student study sessions operated across multiple platforms for three separate courses [1]. Their results found that given multiple education platforms, students will often operate within *platform silos*, or only utilize one educational platform during an individual study session. Of the student sessions, more than 90% of them included only one platform. In a follow-up study, they compared the activities of higher performing students to the lower performing group and showed that both groups were most likely to stick within platform silos [15]. Their work serves as the motivation for our RQ1. Our hypothesis is that the presence of these 'platform silos' will continue to hold across other educational platforms not studied in their research.

3. STUDY

3.1 Design

We studied problem solving in the context of an online professional development course in Python programming. This is a preparatory course for a series in AI that is aimed at non-traditional students making a career transition. The course used the Moodle Learning Management System and TYPOS, a CS exercise platform [14], and is organized into 10 modules Each module includes a set of static reading material, lecture slides, prerecorded videos, optional practice exercises, and a module assessment. There were 24 optional exercises per module, 3 exercises for each of the 8 types previously described. Students were free to work on the practice exercises or assessments as much as they liked. The only requirement for progressing to the next module was to earn a passing grade (80% or higher) on the prior module's assessment. In order to complete the course, students needed to earn passing grades on all assessments.

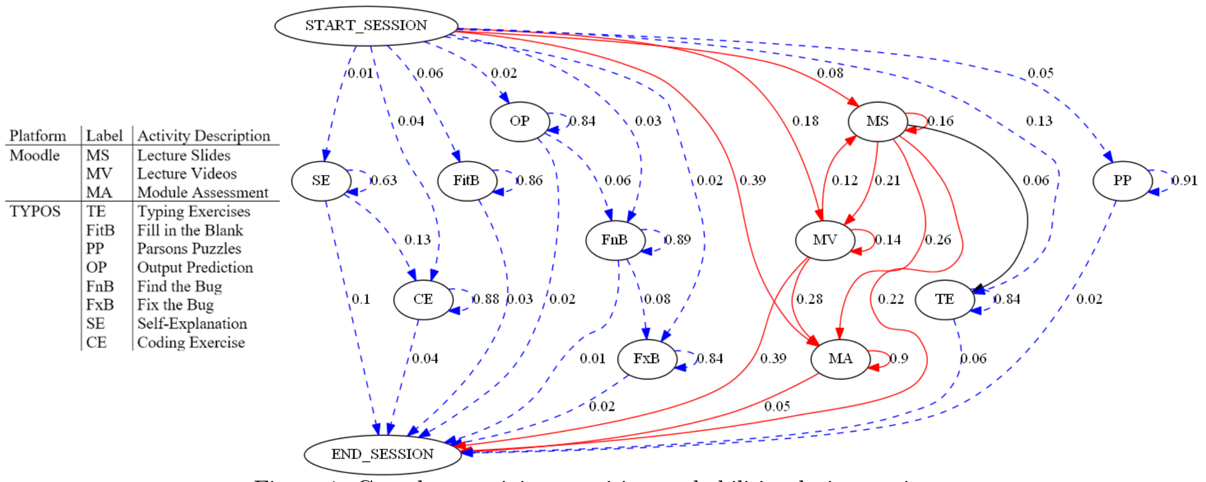
We had 69 students consent to the study. Of those students, 37 successfully completed the course. Student interactions on both platforms were logged. We omitted some Moodle interactions such as like "file downloading" and "viewing the course" which did not pertain to the explicit learning actions we were foucused on.

The resulting dataset contained a total of 29,190 interactions from all the students. We then used a similar strategy as [1] to extract user sessions from these interactions based upon an exploratory analysis of the gaps between interactions. The time deltas between course interactions were measured. If a delta between interactions exceeded a predefined cutoff threshold, that session was considered over, and a new session was created. This was repeated for all interactions a student had for the course. While Seshadri et. al. used a 40 minute cutoff to establish the end and start of a new session, we chose a 60 minutes as it was our most frequently observed delta between interactions. We extracted 1,313 sessions in total. Students that completed the course accounted for 71%, or 20,748, of the course interactions, with 1,041 sessions total, at an average of 28.1 ($\pm$17.9) sessions per completer.

3.2 Activity Session Transition Probabilities

The route that students take through online materials can be modeled as discrete Markov processes, in which each state represents an activity within the session. For example, a student may transition from reviewing lecture slides to viewing lecture videos on Moodle, or $MS \rightarrow MV$. Jeffries et. al. [17] used a similar process to analyze success and help seeking behaviors with students in an introductory CS course.

Figure 1 visualizes the transition probabilities for completers: transitions within TYPOS appear as dashed blue lines, transitions within Moodle are solid red lines, and transitions between TYPOS and Moodle are solid black lines. For visibility, only transitions involving the start/end of a session or those with a frequency above 5% are presented. Module assessment (MA) accounted for 39% of starting session behavior, TYPOS practice accounted for 36%, and lecture slides and videos (Content Consumption) accounted for 26%. This figure shows students' practice was largely siloed by platform with each session taking place within a single mode of interaction. With the exception of the $MS \rightarrow TE$ transition, students either interacted with TYPOS or Moodle, but rarely together. Since MA showed the highest starting session probability, one assumption is that students enrolled in the course with prior coding experience may have reviewed the course material to become familiar with Python syntax before going on to complete the module assessment.



Platform	Label	Activity Description
Moodle	MS	Lecture Slides
	MV	Lecture Videos
	MA	Module Assessment
TYPOS	TE	Typing Exercises
	FitB	Fill in the Blank
	PP	Parsons Puzzles
	OP	Output Prediction
	FnB	Find the Bug
	FxB	Fix the Bug
	SE	Self-Explanation
	CE	Coding Exercise

Figure 1: Completer activity transition probabilities during sessions.

One interesting observation from Figure 1 is that TYPOS practice sessions involved only one or a small subset of the exercise types available to the students. For example, the TE, PP, and $FitB$ exercises were typically completed alone. Among two exercise pairs, $SE \rightarrow CE$, $OP \rightarrow FnB$, and $FnB \rightarrow FxB$ showed higher probabilities than ending the session.

We can further classify the activities by their respective ICAP modes. For example, the $SE \rightarrow CE$ transition can also be viewed as a Constructive→ Interactive transition, $MS \rightarrow TE$ can be viewed as Passive→ Active, and so on. From this perspective, transitions between activities rarely 'downgraded' to a lower mode. With the exception of $MS \rightarrow MA$ and $MV \rightarrow MA$, mode changes primarily shifted by one level of engagement.

3.3 Activity Session Factor Analysis

The probabilities shown in Figure 1 raise new questions about students' practice behaviors. Not only were we able to confirm the existence of *platform silos* from our RQ1, but based on the observed transition probabilities, we found that students may also operate within what we term an *activity silo*, focusing primarily on one ICAP mode per session. In this section we report on two factor analyses which we use to identify the latent variables for each practice session in order to strengthen our claim.

Factor Analysis is a dimension reduction method to describe the variability of observed variables into, potentially, lower latent variables, or factors. To prepare our dataset for factor analysis, each activity was converted into a binary value, representing the presence or absence of the activity in the session. For example, if a session only involves passive content consumption, the resulting vector for the session would be [1,1,0,0,0,0,0,0,0,0,0], where the 1s represent the presence of lecture slides and videos and 0 represents the absence of all other activities.

In order to evaluate the appropriateness of our data for factor analysis, we use the Bartlett's and Kaiser-Meyer-Olkin tests. Bartlett's test compares our correlation matrix against an identify matrix to test whether our samples are from populations with equal variance. Our samples were statistically significant ($\chi^2 = 3360.05, p = 0.0$) and thus we can continue with our factor analysis. Kaiser-Meyer-Olkin checks the adequacy for our variables to determine the suitability of factor analysis. Our KMO score was 0.83, which again shows our dataset is adequate.

The next step for our analysis was to determine the appropriate number of factors. Table 1 shows the eigenvalues for each factor and their cumulative variance. Based on these results, we utilized two separate factor analyses. The first analysis uses 3-factors to correspond to the 3 eigenvalues greater than 1, as suggested by Kaiser [18]. The second analysis increases to 5-factors based on the variance extraction rule, which specifies a 0.7 threshold for eigenvalues [6, 24, 23].

Table 1: Eigenvalues for Factor Analysis of Completers

Factor	Eigenvalue	Cumulative Variance
1	3.860695	30.51%
2	1.390921	37.21%
3	1.170659	41.79%
4	0.916401	52.26%
5	0.810977	62.27%
6	0.665925	60.91%
7	0.649612	69.59%
8	0.486275	70.05%
9	0.446064	55.23%
10	0.391094	55.42%
11	0.211376	55.42%

Our next task was to identify load factor thresholds for our latent variables. While there is no universal standard for loading thresholds, the goal is to only observe variables that share a strong association with each other and is a non-trivial process [22]. For our paper, we will focus our attention to variables above a 0.50 (or 25% of the variable's variance) threshold, highlighting them in our tables as green. Since the difference between a 0.50 and 0.49 loading is minimal, we will also highlight values greater than 0.4 (or 16% variance) in yellow for additional reference.

Table 2 shows the factor load values for our 3-factor analysis. F1 has high loadings for OP, FnB, FxB, SE, and CE. If we consider the ICAP modes for this factor, this indicates a transition of *Constructive* → *Interactive* TYPOS Practice. F2 has high loadings for $FitB$ and PP, or *Active*→ *Constructive* TYPOS Practice. Finally, F3 has high load for MS, or *Passive* Moodle Interaction. From Table 2, we once again can confirm the presence of *platform silos*, however we expand our factor analysis in order to see the presence of *activity silos*. Moreover, 3-factors only accounts for 41.79% of the cumulative variance and so using the 0.7 eigenvalue threshold will allow us to account for 62.27%.

And finally, Table 3 shows the factor load values for our 5-factor analysis. Similar to Table 2, we see a separation between

Table 2: Loadings for 3 Factor Analysis for Completers. Values greater than 0.4 are in yellow and greater than 0.5 are in green.

Activity	F1	F2	F3
MS	0.0306	-0.0294	0.5073
MV	-0.1006	-0.0999	0.3891
MA	-0.0269	-0.2862	0.2177
TE	0.0193	0.4336	-0.1345
FitB	0.4703	0.5471	-0.0286
PP	0.3300	0.6726	0.0392
OP	0.6257	0.3885	0.0216
FnB	0.8215	0.2149	0.0307
FxB	0.8399	0.1546	0.0059
SE	0.6802	0.1020	-0.0933
CE	0.5019	0.0135	-0.1306

TYPOS activity and Moodle activity. Further, the activities for each factor are confined to a single ICAP mode, or within one mode. F1 contains mostly *Constructive* activities (as well as FxB), F2 contains *Active*→ *Constructive* activities, F3 contains *Passive* activities, and F4 and F5 contain *Interactive* activities. In addition, F3 and F4 show a separation between *Passive* Moodle content consumption and *Interactive* assessment taking. Thus, from the results of our 5-factor analysis, we confirm the presence of *activity silos* within our students.

Table 3: Loadings for 5 Factor Analysis for Completers. Values greater than 0.4 are in yellow and greater than 0.5 are in green.

Activity	F1	F2	F3	F4	F5
MS	0.0224	-0.0065	0.2411	0.1061	-0.0150
MV	-0.0836	-0.1009	0.9838	-0.0982	-0.0264
MA	-0.0438	-0.1664	0.1175	0.9759	-0.0124
TE	0.0364	0.3448	-0.0843	-0.2002	-0.0106
FitB	0.4344	0.5629	-0.0451	-0.0445	0.0772
PP	0.2740	0.7467	0.0004	-0.0193	0.0369
OP	0.5520	0.4568	-0.0008	0.0282	0.1792
FnB	0.8419	0.2321	0.0035	-0.0110	0.0829
FxB	0.8759	0.1568	0.0058	-0.0255	0.0980
SE	0.5963	0.1562	-0.0318	-0.0450	0.2602
CE	0.3227	0.0549	-0.0631	-0.0088	0.8891

3.4 Comparing Completers to Non-Completers

Having shown the basic activity structures and identified relevant factors we then chose to explore was the difference between completer and non-completer students. We used the same methods for the non-completer group for comparison. There were 32 students that failed to complete our course. Non-completers made 8,442 course interactions across 341 sessions, with an average 10.7 (±7.8) sessions per non-completer.

We first produced the same transition probabilities diagram for non-completer activity sessions, seen in Figure 2. Similar to completers, module assessment accounted for 31% of starting session behavior, TYPOS practice accounted for 49%, and lecture slides and videos (Content Consumption) accounted for 22%. Non-completers primarily operated within a single platform, though there was more interactions between Moodle and TYPOS. For example, 12% of MV's transitions migrated to TYPOS exercises and 5% of SE transitions migrated to MA. While completer students separated $SE \rightarrow CE$ and $OP \rightarrow FnB \rightarrow FxB$ transitions, these two sequences were combined for non-completers. However, this could potentially be due to the size differences. Both populations had similar population sizes, but non-completers did not complete each module assessment and would not produce as many sessions.

We then carried out the same factor analyses for non-completers. The results of a Bartlett's test showed statistically significant differences ($\chi^2 = 997.8, p > 0.0001$) and our KMO score was also adequate for analysis (0.77). Similar to Table 1, we found support for 3- and 5-factor analysis, seen in Table 4. We note that a 6-factor analysis is also possible, but to mirror the factor analysis for completers, we elected not to pursue it.

Table 4: Eigenvalues for Factor Analysis of Non-completers

Factor	Eigenvalue	Cumulative Variance
1	3.480694	26.26%
2	1.503161	34.57%
3	1.286916	41.39%
4	0.888718	47.44%
5	0.821608	54.51%
6	0.719444	62.12%
7	0.657629	66.31%
8	0.520019	63.79%
9	0.499220	57.46%
10	0.375677	57.69%
11	0.246915	57.69%

Table 5 shows our 3-factor analysis for non-completers. The same activities having high loadings as F1 and F2 as the 3-factor analysis for completers (Table 2) and also show similar *platform silos*. Likewise, the ICAP mode considerations are similar for each factor. F1 shows *Constructive*→ *Interactive* behaviors, F2 shows *Active*→ *Constructive* behaviors, and F3 shows *Passive* Moodle interaction.

Table 5: Loadings for 3 Factor Analysis for Non-completers.

Activity	F1	F2	F3
MS	0.0362	0.0014	0.4167
MV	-0.0334	0.0106	0.6616
MA	-0.0136	-0.2524	0.2999
TE	0.0756	0.4637	-0.0719
FitB	0.3176	0.6024	-0.0202
PP	0.1082	0.7404	0.0221
OP	0.5310	0.3499	0.1105
FnB	0.5573	0.4837	-0.0910
FxB	0.6636	0.3369	-0.1218
SE	0.8012	0.0568	0.0573
CE	0.5900	0.0201	0.0056

Table 6 shows our 5-factor analysis for non-completers. Non-completers maintained the *Constructive*→ *Interactive* connection for F1 and F2 also maintains the *Active*→ *Constructive* connection. The remaining factors do differ, F3 separated the $FnB \rightarrow FxB$ exercises from F1 and F4 focuses primarily on TE. The absence of MA was expected since course progression requires passing module assessments. From our analysis, we conclude that non-completers still operated within *activity silos.*

Table 6: Loadings for 5 Factor Analysis for Non-completers.

Activity	F1	F2	F3	F4	F5
MS	0.0271	0.0234	0.0080	-0.0129	0.4040
MV	-0.0191	0.0204	-0.0417	0.0373	0.7123
MA	0.0049	-0.1705	-0.0558	-0.1637	0.2922
TE	0.0541	0.2612	0.0605	0.9570	-0.0716
FitB	0.2259	0.6333	0.1698	0.1275	-0.0557
PP	0.0224	0.6734	0.1463	0.1925	-0.0155
OP	0.4707	0.4584	0.1711	0.0080	0.0735
FnB	0.2516	0.4546	0.6255	0.0378	-0.0543
FxB	0.3618	0.2049	0.8444	0.0752	-0.0706
SE	0.8072	0.0883	0.2525	0.0740	0.0496
CE	0.6201	0.1006	0.1092	-0.0054	-0.0206

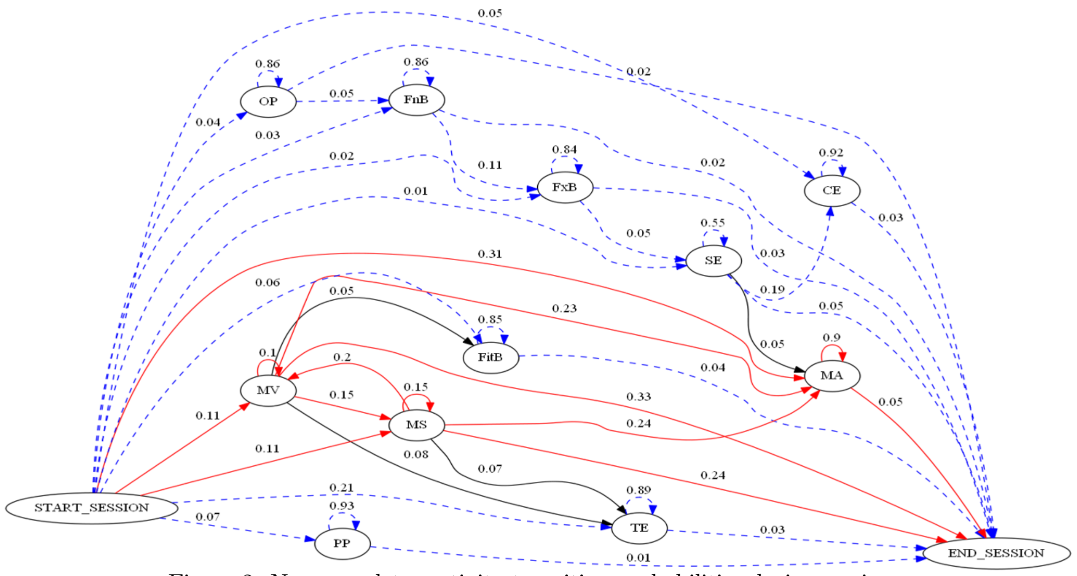


Figure 2: Non-completer activity transition probabilities during sessions.

4. DISCUSSION

The results from our probability transition diagrams confirm the presence of both *platform silos* and *activity silos* in student work. They also serve to highlight areas where educators and researchers can tailor more appropriate learning paths for students and in particular those students that may be struggling with course material.

Our study allowed students to self-select which activities they wanted to focus their attentions on. While this style of course design could be adopted, it does still present limitations. However, students that primarily focus on lower-level ICAP mode activities may be reluctant to move into high-level modes. Instructors or systems that can identify this stagnate practice behavior could encourage students to move into high-level ICAP modes. There is growing interest in the concept of *nudge theory* to "alter behavior without incentives or banning alternatives" [7] to encourage progression.

Similarly, we presented students with a number of different activities, at different complexities, for their learning experience. Based on our results, students were more than willing to complete each type of exercise. Some students even asked for more activities in our post-course survey. While increasing the workload for students and learning material creators, many of the activities we used are not overly complex and required a minimal amount of time to create, or from the students' perspectives complete. Activities like typing exercises or Parsons puzzles can be created from existing course materials and offer little incentive for students to cheat. They simply allow students an opportunity to practice the concepts they learned rather passively given to them, refining their understanding, before needing to apply it to problem solving activities like coding exercises.

5. LIMITATIONS

We acknowledge some limitations with our study. First, our course ran during the COVID-19 pandemic, which has altered many individuals' habits. Our population also contained non-traditional students who were balancing their studies with working from home and supporting other family members. Thus, non-completion may have been driven by external constraints that are not reflected in our dataset, and the observed habits may change somewhat during non-COVID times.

Second, the exercise types were presented in a consistent manner for each module. Thus they were implicitly sequenced with lower-level ICAP modes appearing on the top. As we mentioned in our introduction, discerning the appropriate order for 11 different activities is a non-trivial matter and measuring the appropriate order of exercise types was not a part of our study. Thus, we presented exercises in an order that progressively increased the level of engagement. This may have influenced next practice selections by students.

Finally, we acknowledge that the ICAP modes associated with each exercise type are somewhat subjective and open for discussion. Moreover the exact evaluation of exercises like Parsons Puzzles or self-explanation may require additional research and context. For the purposes of this study, when faced with uncertainty we classified exercises according to a higher level mode of interaction.

6. CONCLUSIONS

In this work, we extracted the practice and study session behaviors from non-traditional students learning Python. Among completers and non-completers of the course, they primarily focused on a single platform. The activities within these platforms were mapped to the ICAP framework. Further, we used factor analyses to identify the presence of *activity silos* within practice sessions. Completers and non-completers shared similar behaviors during these practice sessions, primarily focusing on one or two modes of engagement and rarely 'downgraded' to lower level modes.

We can utilize these activity sequences to help shape our overall course designs for ensuring student learning. Lower-level activities can provide students with the foundational knowledge necessary as a part of the technical skills for the content, while higher-level activities can refine and encourage additional learning gains. As the research in this area expands, we hope the information presented in this study encourages educators and researchers alike to provide practice in both levels and can serve as a guide for recommendations on how to best build long-term proficiencies.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant DRL 1721160. Kristy Boyer, Eric Wiebe, and Collin F. Lynch (co-PIs).

7. REFERENCES

[1] Sheshadri Adithya, Niki Gitinabard, Collin F Lynch, Tiffany Barnes, and Sarah Heckman. Predicting student performance based on online study habits: A study of blended courses. *International Educational Data Mining Society*, 2018.

[2] Joe Michael Allen, Frank Vahid, Alex Edgcomb, Kelly Downey, and Kris Miller. An analysis of using many small programs in cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 585–591, 2019.

[3] A. Altadmri and N. Brown. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 522–527, New York, NY, USA, 2015. ACM.

[4] Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. Learning from examples: Instructional principles from the worked examples research. *Review of educational research*, 70(2):181–214, 2000.

[5] Robert K Atkinson and Alexander Renkl. Interactive example-based learning environments: Using interactive elements to encourage effective processing of worked examples. *Educational Psychology Review*, 19(3):375–386, 2007.

[6] Deborah L Bandalos and Sara J Finney. Exploratory and confirmatory. *The reviewer's guide to quantitative methods in the social sciences*, 93, 2010.

[7] Chris Brown. Digital nudges for encouraging developer actions. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 202–205. IEEE, 2019.

[8] Nick Cheng and Brian Harrington. The Code Mangler: Evaluating coding ability without writing any code. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 123–128, 2017.

[9] Michelene TH Chi. Private Communication, August 2020.

[10] Michelene TH Chi and Ruth Wylie. The ICAP framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243, 2014.

[11] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*, pages 113–124. ACM, 2008.

[12] John Edwards, Joseph Ditton, Dragan Trninic, Hillary Swanson, Shelsey Sullivan, and Chad Mano. Syntax exercises in CS1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, pages 216–226, 2020.

[13] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, pages 20–29, 2017.

[14] Adam M Gaweda, Collin F Lynch, Nathan Seamon, Gabriel Silva de Oliveira, and Alay Deliwa. Typing exercises as interactive worked examples for deliberate practice in CS courses. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 105–113, 2020.

[15] Niki Gitinabard, Tiffany Barnes, Sarah Heckman, and Collin F. Lynch. What will you do next? A sequence analysis on the student transitions between online platforms in blended courses. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*, 2019.

[16] Kyle James Harms, Jason Chen, and Caitlin L Kelleher. Distractors in parsons problems decrease learning efficiency for young novice programmers. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, pages 241–250, 2016.

[17] Bryn Jeffries, Timothy Baldwin, Marion Zalk, and Ben Taylor. Online tutoring to support programming exercises. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 56–65, 2020.

[18] Henry F Kaiser. The application of electronic computers to factor analysis. *Educational and psychological measurement*, 20(1):141–151, 1960.

[19] Antti Leinonen, Henrik Nygren, Nea Pirttinen, Arto Hellas, and Juho Leinonen. Exploring the applicability of simple syntax writing practice for learning programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 84–90. ACM, 2019.

[20] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*, pages 101–112. ACM, 2008.

[21] Dale Parsons and Patricia Haden. Parson's programming puzzles: A fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, pages 157–163, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[22] Robert A Peterson. A meta-analysis of variance accounted for and factor loadings in exploratory factor analysis. *Marketing letters*, 11(3):261–275, 2000.

[23] Keenan A Pituch and James P Stevens. *Applied multivariate statistics for the social sciences: Analyses with SAS and IBM's SPSS*. Routledge, 2015.

[24] John Ruscio and Brendan Roche. Determining the number of factors to retain in an exploratory factor analysis using comparison data of known factorial structure. *Psychological assessment*, 24(2):282, 2012.

[25] Susan Wiedenbeck, Vikki Fix, and Jean Scholtz. Characteristics of the mental representations of novice and expert programmers: an empirical study. *International Journal of Man-Machine Studies*, 39(5):793–812, 1993.

[26] Greg Wilson. *Teaching Tech Together: How to Make Your Lessons Work and Build a Teaching Community around Them*. CRC Press, 2019.

[27] Guojing Zhou, Jianxun Wang, Collin F Lynch, and Min Chi. Towards closing the loop: Bridging machine-induced pedagogical policies to learning theories. *International Educational Data Mining Society*, 2017.